TP OPTIMISATION 2

Supprimer les tables EMP et DEPT présentes dans votre schema.
Exécuter le script TpBuild.sql présent dans le repertoire :
\\documents\is3+4\sgbd\LEKPA\TpBuild.sql

Supprimer tous les index que vous avez créés précédemment.
Pour identifier les index que vous avez créés :
> select **index_name** from user_indexes
where table_name = 'EMP' and index_name not like 'SYS%' ;
Pour supprimer les index : drop index <**index_name**> ;
Pensez à collecter les statistiques des tables/index à chaque modification.

Exercice 1 :
Optimisez la requête suivante :
SQL> SELECT * FROM EMP WHERE JOB=’Clerk’ and DEPTNO=20;

1- Créer un index sur la colonne JOB et un index sur la colonne DEPTNO. **Collectez les statistiques**. Vérifier le plan d’exécution.
2- Créer un index ayant pour clés les deux colonnes JOB et DEPTNO. **Collectez les statistiques**. Vérifier le plan d’exécution
3- Exécuter cette requête 10 fois :

SQL> INSERT INTO EMP SELECT * FROM EMP; COMMIT;

Vous obtiendrez alors 14336 lignes

Supprimer les index précédemment créés et répétez les points 1 et 2

Exercice 2 :
Supprimer tous les index précédemment crées.
Optimiser la requête suivante :
SQL> select deptno, sum(sal)from EMP where deptno = 30 and job = 'SALESMAN' group by deptno;

Exercice 3 :
Supprimer tous les index précédemment créés
Optimiser la requête suivante :
SQL > select deptno, sum(sal) from EMP
where deptno in (select deptno from DEPT where loc = 'DALLAS' or loc = 'CHICAGO')
group by deptno;

Exercice 4

1.1/ Quel est le résultat de la requête suivante :

select deptno from dept where deptno not in (select deptno from emp);

1.2/ Récrire la requête SQL sous forme :

- not exists
- jointure
- ensembliste (MINUS)

Comparer les plans d'exécutions (chemin, coût, blocs, tris et temps écoulé)

1.3/ Proposer des optimisations pour optimiser la requête

Exercice 5 :

Créer une table BIGBITMAP ayant trois colonnes id, nbitmap et date_insertion. Exécuté le script ci-dessous **3 fois** pour insérer les lignes dans la table.

```
BEGIN
    for id in 1..100000 LOOP
     INSERT INTO BIGBITMAP VALUES
       (  id,
          MOD(id,5) ,
         TO_DATE(sysdate, 'DD/MM/YYYY')
       );
     Commit;
  END LOOP;
  dbms_output.put_line ('100000 lignes inserees');
END;
/
```

Le champs nbitmap aura une faible cardinalité (5 pour cet exemple : nbitmap contient 0,1,2,3,4)

Créer un index btree sur la colonne nbitmap.
Générer le plan d'exécution et calculer le temps d'exécution avec 'set timing on'.

Puis créer un index bitmap sur la colonne nbitmap.
Générer le plan d'exécution et calculer le temps d'exécution avec 'set timing on'.

Comparer et expliquer les résultats (temps d'exécution et plan d'exécution) sur les ordres suivants :

```
select count(*) from bigbitmap where nbitmap = 0 ;
select sum(indice) from bigbitmap where nbitmap = 0 ;
```